(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 398 645 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
06.08.1997 Bulletin 1997/32

(51) Int Cl.[6]: **G06F 17/30**, G06F 1/00

(21) Application number: **90305218.1**

(22) Date of filing: **15.05.1990**

(54) System for controlling access privileges

System zur Steuerung von Zugriffsprivilegien

Système pour contrôler des privilèges d'accès

(84) Designated Contracting States:
DE FR GB

(30) Priority: **15.05.1989 US 352081**

(43) Date of publication of application:
22.11.1990 Bulletin 1990/47

(73) Proprietor: **International Business Machines Corporation**
Armonk, N.Y. 10504 (US)

(72) Inventor: **Fabbio, Robert Anthony**
Austin, TX 78759 (US)

(74) Representative: **Burt, Roger James, Dr.**
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

(56) References cited:

- **1989 IEEE COMPUTER SOCIETY SYMPOSIUM ON SECURITY AND PRIVACY 3 May 1989, OAKLAND, CALIFORNIA pages 110 - 115 E.B.FERNANDEZ ET AL. 'A security model for object oriented databases'**

EP 0 398 645 B1

Printed by Jouve, 75001 PARIS (FR)

Description

This invention relates to a system for controlling access privileges to data within a data processing system.

In a relational database, data is viewed as rows and columns, where a row represents a record, in tables. In order to retrieve records that represent the spanning of multiple tables, relational operators, such as a join operation, are utilised. Although relational operators are used within applications or by users through a query language, these relational operations are not incorporated within the data such that the relational database manager can automatically intuit such relationships. In order to retrieve a record from the database, the user must have read permission for that record. In order to change the data in the record, the user must retrieve (select and fetch) the data into memory, update the record in memory, and then write the updated record to the database. The user must have permission to write the record before the database manager allows the previous data to be overwritten in the database. Moreover, in order to update the record in memory, the user must be cognizant of the column's attributes, such as whether that column contains integers, characters, strings etc.

In an object oriented database, the navigation through the data is similar to that of the network model where an object oriented database manager traverses the relationships defined amongst the objects and subobjects. In contrast to the relational database model, relationships between objects and subobjects are defined through the data, and not through explicit operations. Moreover, in complex application environments, objects and their relationships usually show very complex internal structures and comprise larger numbers of properties; such properties include methods. In addition, the object oriented database model parallels the object oriented programming model in that one may inherit attributes and methods from predecessor objects. However, the object oriented programming model generally deals with objects that are temporal in nature, i.e., memory resident. The object oriented database model extends the programming model by allowing one to deal with objects that persist, i.e., are disk resident. One of the features of object oriented databases (analogous to the programming model) is to provide the ability to define generic operations, i.e. methods, which apply to the objects, for the purposes of manipulating and retrieving them. In the relational database model, all operations to retrieve and manipulate records are performed by using the database manager's application programming interface or query language. The object oriented methodology insulates the users of the database from the data representation and/or data structures that comprise objects. All retrieval and update operations may be performed through the use of these methods.

In the relational database model, access privileges may be set on the records such that grant and revoke authorisation privileges can be determined per user of the database. The records are tagged with either read or write privileges for specific users. For example, the first record could be read by users A, B, and C, and only written by user C. Likewise, the second record could be tagged such that only user A has the permission to read, and only user A has the permission to write.

For example, if a database contained payroll information, members of the payroll department may be able to read all of the payroll information for all departments except the payroll record for their payroll department. Only the manager of the payroll department would have the permission to read the record containing the payroll department data. If a user attempted to retrieve the records which did not define read privilege for the user, that user would not be granted the ability to see that record of data.

One shortcoming of relational databases is that access permission control is on a record basis.

1989 IEEE Computer Society Symposium on Security and Privacy, May 1-3, 1989, Oakland, California, pages 110-115 describes an authorization model for object oriented databases in which authorization rules are placed at various parts of an object hierarchy and evaluated against access requests using an algorithm which searches the object hierarchy for a rule authorising the requested access.

This invention provides an object oriented database system as defined in the claims.

An embodiment of the invention will be described in the ensuing description with reference to the following figures, wherein

Fig. 1 is a block diagram of the system of this embodiment having an object data manager for controlling the access to objects in an object database according to an access control policy associated with the objects and the user's credentials.

Fig. 2A is a hierarchy of objects in an object database showing superobjects, objects, and subobjects.

Fig. 2B is an illustration of an access control list for each object in an object database.

Fig. 2C is a hierarchy of interface menu objects in and object database showing superobjects, objects, and subobjects.

Fig. 3A is a flow diagram of the object data manager utilising the access control facilities to determine whether a particular operation against a set of object classes and objects are permitted.

Fig. 3B is a flow diagram of the methodology for granting or revoking the operations based on the access control policies.

Fig. 3C is a continuation of the flow diagram of Fig. 3A.

Fig. 4 illustrates the structure of the access control list.

Fig. 5A shows the access control policy of this embodiment used in conjunction with a system management interface tool for controlling the administrative views of menu objects.

Fig. 5B shows a hierarchy of system management interface menu option objects.

The data processing system 1 has an operating system running in memory 3 which has tables 30 which contains the credentials 31 for each active user 32. The credentials identity 31 contains the user id 33, and the group ids 34 for which the user belongs and other security information 35. In user memory 4, the object data manager 40 accesses the objects found in the file system. The file system 20 resides on disk 2 which has object classes 50, as shown in Fig. 2A and Fig. 2C.

Referring to figure 2A, object class 54 is a super class of objects 55 and objects 56, and objects 55 is a superclass of objects 57. An example is described in EP-A-0 398 643. Likewise, objects 55 are a subclass of object class 54, and object classes 57 are subobjects of object class 55. The object class of the system management interface tool comprises interface menu option objects, as shown in Fig. 2C. An example of such an interface tool is described in EP-A- 398 646.

Referring back to Fig. 1, the object data manager 40 is cognizant of absolute access control policy assignment and authorisation per object. The authorisation process reviews the user's credentials 31 described to the object data manager 40 and uses this information to determine the operations which are permissible on each object.

When the object data manager is requested to perform operations on specific objects, i.e. retrieve or modify, the object data manager first determines the current credential attributes 31 for the user 32 requesting the operation by interfacing with the kernel 3. The object data manager then performs the operation which may require inheritance or subclass traversal through the objects. As the object data manager performs such an operation, the access control policies lists are checked for each object satisfying the criteria of the operation. Access control policies are treated like other attributes within an object class in that they may be inherited from other superclasses, thus altering the access control policies as traversal of the objects is performed.

Fig. 2B represents a user defined view of the devices object class 54 and one of its subclasses, customised devices 55. Devices object class 50 contains attributes 71, methods 72 and triggers 73. For example, for an object class of customised devices devices 55, the attributes 71 would represent the definition of the object such as device name 74 character string, device id 75 integer number, and device type 76 character string, etc. The methods 72 are the operations that apply to the object. For example, a method of configuring the device 77, the method of defining the device 78, the method of unconfiguring 79, and the method of undefining 80 the object, etc. Triggers 73 represent the events that are automatically invoked when the objects are manipulated.

In addition to the user defined attributes 71, the object data manager transparently maintains a separate access control list 100 as part of each object within each object class. The access control list is further shown in Fig. 4. The access control list 100 maintains the owning user id 101 and the owning group id 102 for each object. This owning information 101, 102 dictates the access control policy for maintenance to the access control list itself. Only those owning users and owning groups can alter the access control list in an object. Before any access control entry is altered for any object, the object data manager first verifies that the user can be identified through its credentials in either the owning user id 101 or the owning group id 102.

The access control attributes on each object consists of eight 32 bit entries 111-118, of which seven of these entries 111-117 represent the user or group ids making up the access control list 100. The eighth entry 118 is divided into eight 4-bit slots 121-128, where the first seven slots 121-127 represent the privileges associated with the corresponding access control entry 111-117. The eighth 4-bit slot 128 is used to keep the count of the number of entries used. In the first seven 4-bit slots 121-127, the first three bits represent read, write, and execute privileges while the last bit of the 4-bit slot indicates whether the corresponding access control entry applies to a user or a group id.

Fig. 3A illustrates the high level flow of the object data manager utilising the access control facilities to determine whether a particular operation against a set of object classes and objects are permitted. The object data manager opens the appropriate object classes, step 301, and determines if the open succeeded, step 302. The object data manager then iterates for each object class that was opened, step 303. When the list of object classes opened are processed, the results are accumulated and returned to the user, step 304. If there are object classes to process, the object data manager acquires the user's credentials from the operating system (resultant is the least privilege associated for that user based on the set's credentials), step 305. The object data manager then performs the retrieval of the selected objects, step 306, and checks to see if the object is available for further processing, step 310, specifically whether the access privileges for the user do not conflict with the access controls assigned to the object being accessed. Specifically, the object data manager checks in the operating system for the credentials for the user, and checks these credentials with the list of access control

privileges defined by the objects retrieved. If the access privileges are denied, the object data manager then checks to see if there are more objects that meet that selection criteria, step 315. If so, the object data manager returns to step 305. If not, the object data manager returns to step 303. Given that the object holds the appropriate access controls for further processing, step 320, the object data manager performs the specific operations generally defined by the methods. The object data manager accumulates the new set of access control information for the user based on the bitwise ANDing of the object's access controls, step 325 and returns to step 315 to determine whether there are more objects to retrieve. In step 325, the object data manager uses this technique to inherit the least access privilege that spans the objects of interest.

Fig. 3B describes in more detail step 310 from Fig. 3A which describes the methodology for granting or revoking the operations based on the access control policies. The access control facility first determines whether the requested modes supplied to it are valid, step 328. Given that the modes are valid, the object data manager defines a bit mask which is representative of the requested modes, step 331. In addition, the object data manager accesses the in memory version of the access control list assigned to a particular object, step 335. Once the access control list has been acquired, the object data manager determines the number of entries utilised within the list, step 337. If the count is not greater than zero, step 331, the object data manager grants access to the object by the user, step 341. If the count is greater than zero, the object data manager compares the user's credentials to the access control entry representing either a user or a group, step 344. If there is a match, the object data manager saves the index of the matching access control entry, step 346. It then checks to see if there are additional access control entries, step 352, and if so, returns to step 344. If additional entries do not exist, the object data manager determines if there were any matches between the user's credentials and the object's access control entry, step 355. If there are none, the access is denied to the object for that user, step 357. If a match exists, the object data manager acquires the privilege bit mask defined for that user on that object, step 359, Fig. 3C. The object data manager then checks to see if the requested bit mask is greater than the user's bit mask defined by the access control entry within that object, step 361. If the requested bit masks is greater, then access is granted to the object for that user, step 363. If the requested bits are less than or equal to, then a bitwise AND operation is performed between the requested bit mask and the bit mask found in the objects entry for that user, step 365. If the resultant is greater than zero, step 367, then access is granted to the object for that user, step 369. If the resultant is less than or equal to zero, then access is denied to that object for that user, step 372.

If a particular user is associated with multiple groups, and identifies oneself with multiple groups when performing operations on the objects, the object data manager will perform a bitwise AND operation of the particular sets of credentials that are currently associated with that user. This results in a assigning the least privilege associated with the intersection of the credentials sets.

The above described access control policies can be applied to various applications. An example of such an application is described in EP-A-398 646. The access control lists of this embodiment can be applied to a system management interface tool for the purposes of defining the authorisation policies for the various views that may be accessed by a collection of system administrators with various authorities. Without this embodiment, one approach would be to define the collection of menus, dialogs, and prompts to represent the permutations of the various administrative views associated with the different administrative privileges. However, this technique typically results in very large databases containing a great deal of redundant information.

In contrast, with the present embodiment, the various menus, dialogs, and prompts are only stored once in the object database, and are assigned the appropriate access controls on the various objects (menus, dialogues, prompts) such that the various permutations of administrative views are dictated by the access control policies.

For example, Fig. 5B shows a hierarchy of interface menu option objects for performing system management tasks, 500. Along with security and users, 504, other subclasses of system management interface objects include devices, TCP/IP, physical and logical storage, communications applications and services, problem determination, etc. As shown in Fig. 5A, administrator A, 551, may have read, write, and execute privileges for the user interface objects 553 which present the configurable domain of TCP/IP, devices, and users while administrator B, 552, may have read, write, and execute privileges for the user interface objects which present the configurable domain of just users, 554. However, administrator A, 551, may only have the privilege to access to view the users 542 and the groups 543, while administrator B, 552, has access to view and manage the access control lists, 541, the users 542, the groups, 543, and passwords, 544. Therefore, administrator A would only have access to menus 542, 543, while administrator B would have access to menus 542, 543, 544, 541 as the subobjects of the configurable domains menu object class for users.

There has been described a system in which grant and revoke permissions, referred to herein as access control lists apply to a collection of objects in an object oriented database. An object data manager supports composite objects which are multiple objects and the access control policies that apply to such objects are transparent to the user. The access control lists span across the objects. Furthermore, not only do the access control

lists provide read and write permissions, but they also provide execution permission for operations which apply to the objects. The execute semantics apply to methods that are invoked to perform operations on the objects. A user with execute permission can perform operations on those objects.

For example, if the objects in the object oriented database contained information on the configuration of a data processing system, a user could be either granted or revoked the permission not only to read the configuration data, but also to perform configuration operations on the data. If the configuration information contained information on the physical devices in the system, the user would be granted or revoked the permission to configure, define, start, stop, unconfigure, and undefine those devices.

Claims

1. An object oriented database system comprising a hierarchy of objects (54, 55, 56, 57, 58) and an object data manager (40) for accessing the objects, characterised by one or more tables (30) containing for each active user of the system, credentials which allocate the user to a user id and to one or more user group ids and in that each object within each class has, as an attribute which may be inherited from object superclasses, an access control list (100) for identifying by user id and group id users having access privileges for maintaining the access control list of the object and for identifying operations users are authorised to perform, the object data manager being arranged to determine whether an operation on an object is permissible for a particular user by comparison of ids in the access control list of the object with the ids stored in the credentials for the user.

2. A system as claimed in claim 1 wherein the operations include read, write and execute operations.

3. A system as claimed in claim 1 or claim 2 wherein object data manager is arranged to determine the least amount of privilege associated with a composite object accessed by a user.

Patentansprüche

1. Objektorientiertes Datenbanksystem, das eine Hierarchie von Objekten (54, 55, 56, 57, 58) und einen Objektdatenverwalter (40) zum Zugreifen auf die Objekte umfaßt, gekennzeichnet durch eine oder mehrere Tabellen (30), die für jeden aktiven Benutzer des Systems Berechtigungsnachweise enthalten, die den Benutzer einer Benutzer-ID und einer oder mehreren Benutzergruppen-IDs zuordnen, und dadurch, daß jedes Objekt innerhalb jeder Klasse als Attribut, das von Objektoberklassen vererbt sein kann, eine Zugriffssteuerliste (100) hat, um mit Hilfe von Benutzer-ID und Gruppen-ID Benutzer mit Zugriffsprivilegien zur Pflege der Zugriffssteuerliste des Objektes zu identifizieren und um Operationen zu identifizieren, zu deren Ausführung Benutzer berechtigt sind, wobei der Objektdatenverwalter so eingerichtet ist, daß er erkennt, ob eine Operation an einem Objekt für einen speziellen Benutzer zulässig ist, indem er IDs in der Zugriffssteuerliste des Objektes mit in den Berechtigungsnachweisen für den Benutzer gespeicherten IDs vergleicht.

2. System, wie es in Anspruch 1 beansprucht wird, wobei die Operationen Lese-, Schreib- und Ausführungsoperationen entfallen.

3. System, wie es in Anspruch 1 oder Anspruch 2 beansprucht wird, in dem der Objektdatenverwalter so eingerichtet ist, daß er mindestens den Grad des Privilegs erkennt, das mit einem zusammengesetzten Objekt verbunden ist, auf das ein Benutzer zugreift.

Revendications

1. Système de base de données orienté objet, comprenant une hiérarchie d'objets (54, 55, 56, 57, 58) et un gestionnaire de données objet (40), destiné à l'accès aux objets, caractérisé par une ou plusieurs tables (30) contenant, pour chaque utilisateur actif du système, des références allouant l'utilisateur à des données d'identification, id, de l'utilisateur utilisateur et à un ou plusieurs id de groupes d'utilisateurs, et en ce que chaque objet, dans chaque classe, a, à titre d'attribut pouvant être hérité de superclasses objet, une liste de commandes d'accès (100), pour identifier des utilisateurs par l'id utilisateur et l'id de groupe, ayant des privilèges d'accès pour conserver la liste de commande d'accès de l'objet et pour identifier des opérations que les utilisateurs sont autorisés à effectuer, le gestionnaire de données objet étant agencé pour déterminer si une opération sur un objet est autorisée pour un utilisateur particulier, par comparaison de ses id, se trouvant dans la liste de contrôles d'accès de l'objet, avec les id stockés dans les références destinées à l'utilisateur.

2. Un système selon la revendication 1, dans lequel les opérations comprennent la lecture, l'écriture et l'exécution d'opérations.

3. Un système selon la revendication 1 ou la revendication 2, dans lequel un gestionnaire de données

objet est agencé pour déterminer la dernière quantité de privilège associée à un objet composite ayant subi un accès par un utilisateur.

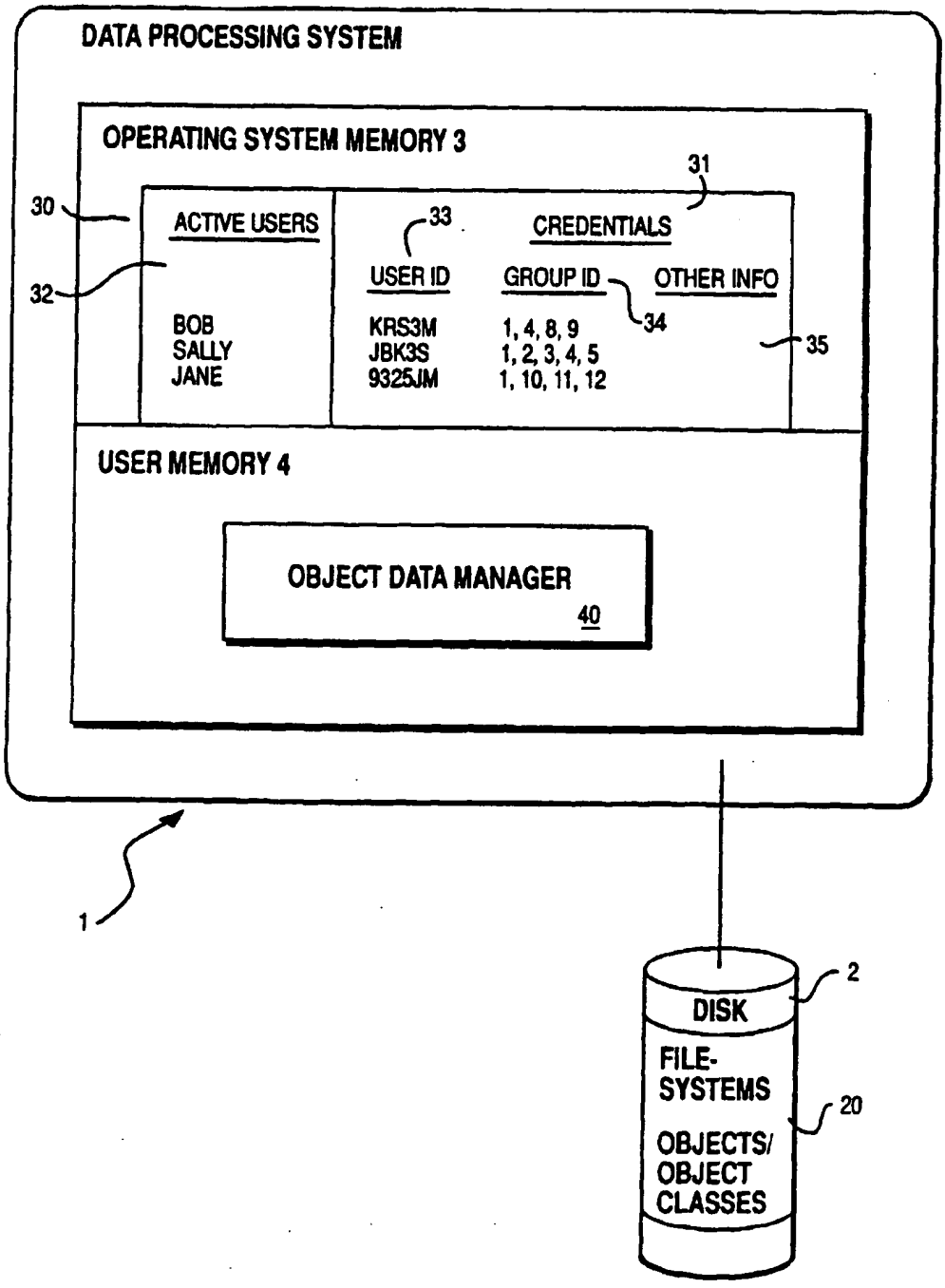
DATA PROCESSING SYSTEM
OPERATING SYSTEM MEMORY 3
31
30
ACTIVE USERS
33
CREDENTIALS
32
USER ID
GROUP ID
OTHER INFO
34
BOB
SALLY
JANE
KRS3M
JBK3S
9325JM
1, 4, 8, 9
1, 2, 3, 4, 5
1, 10, 11, 12
35
USER MEMORY 4
OBJECT DATA MANAGER
40
1
2
DISK
FILE-
SYSTEMS
20
OBJECTS/
OBJECT
CLASSES

FIG. 1

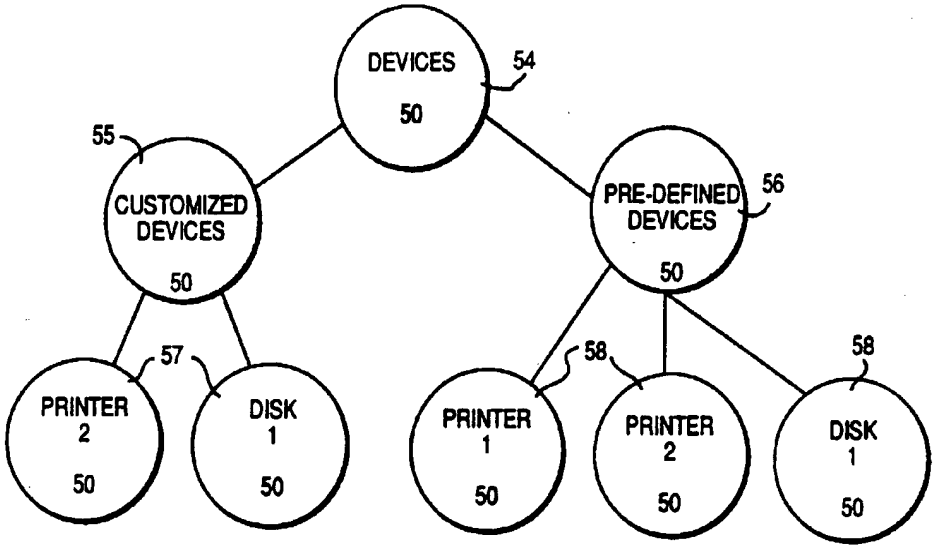
DEVICES
50
54
55
CUSTOMIZED DEVICES
50
PRE-DEFINED DEVICES
50
56
57
PRINTER 2
50
DISK 1
50
58
PRINTER 1
50
PRINTER 2
50
58
DISK 1
50

FIG. 2A

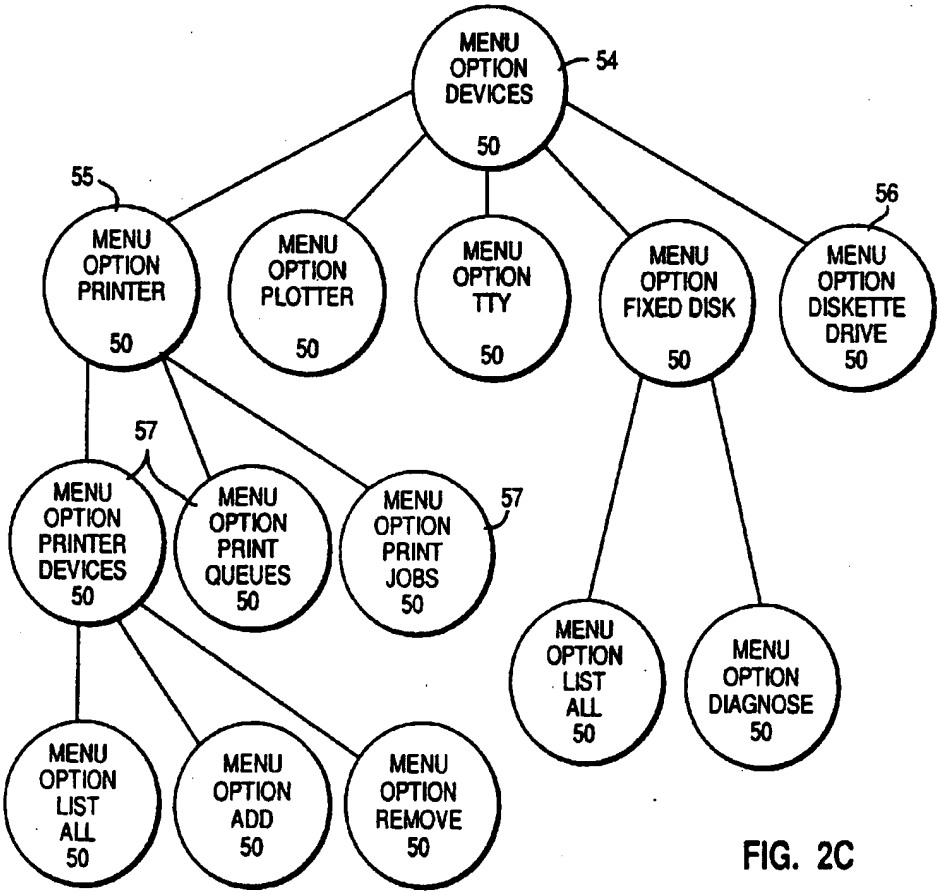
MENU OPTION DEVICES
50
54
55
MENU OPTION PRINTER
50
MENU OPTION PLOTTER
50
MENU OPTION TTY
50
MENU OPTION FIXED DISK
50
56
MENU OPTION DISKETTE DRIVE
50
57
MENU OPTION PRINTER DEVICES
50
MENU OPTION PRINT QUEUES
50
MENU OPTION PRINT JOBS
50
57
MENU OPTION LIST ALL
50
MENU OPTION DIAGNOSE
50
MENU OPTION LIST ALL
50
MENU OPTION ADD
50
MENU OPTION REMOVE
50

FIG. 2C

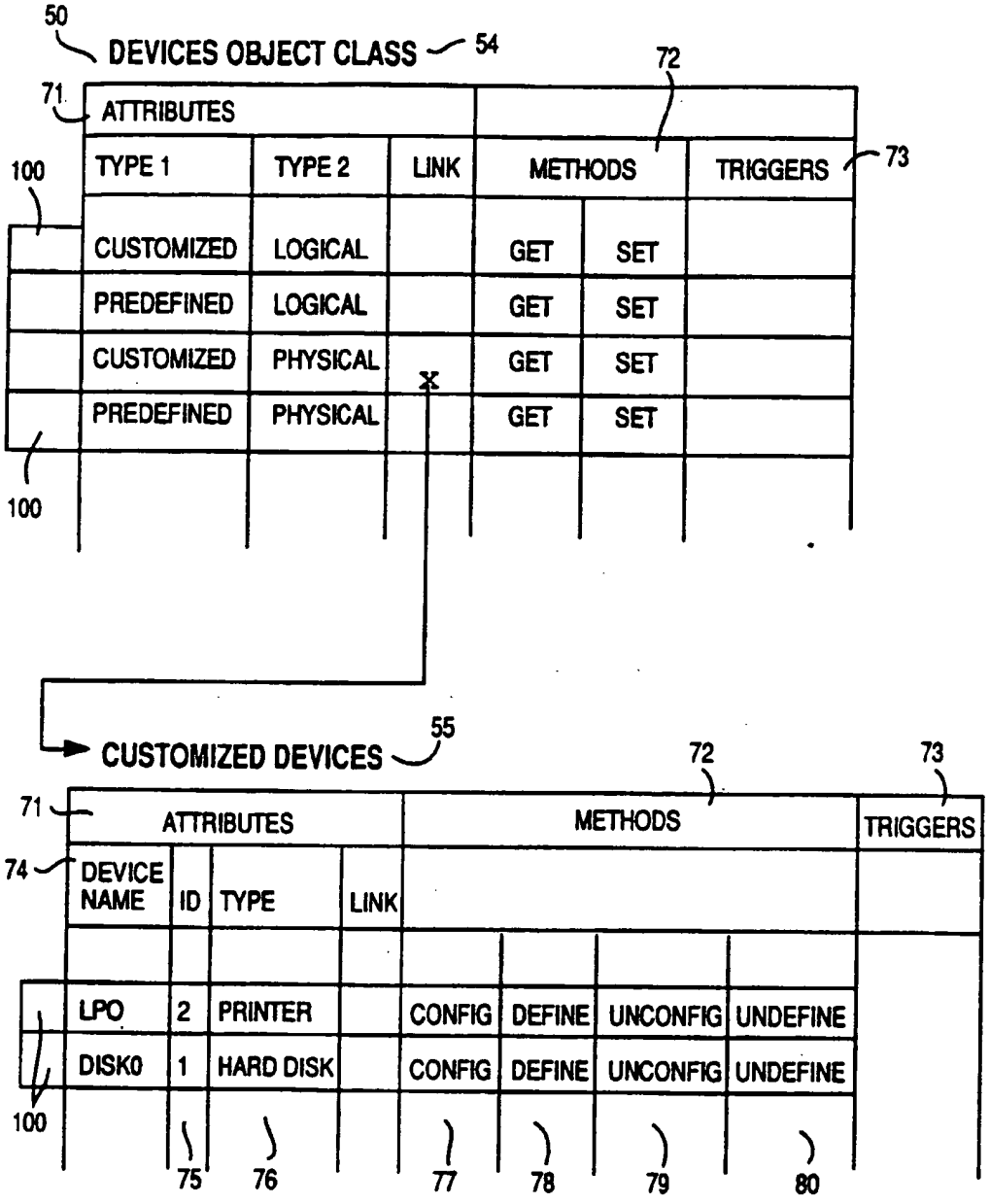
50
DEVICES OBJECT CLASS
54
72
71
ATTRIBUTES
TYPE 1
TYPE 2
LINK
METHODS
TRIGGERS
73
100
CUSTOMIZED
LOGICAL
GET
SET
PREDEFINED
LOGICAL
GET
SET
CUSTOMIZED
PHYSICAL
GET
SET
PREDEFINED
PHYSICAL
GET
SET
100
55
CUSTOMIZED DEVICES
72
73
71
ATTRIBUTES
METHODS
TRIGGERS
74
DEVICE NAME
ID
TYPE
LINK
LPO
2
PRINTER
CONFIG
DEFINE
UNCONFIG
UNDEFINE
DISK0
1
HARD DISK
CONFIG
DEFINE
UNCONFIG
UNDEFINE
100
75
76
77
78
79
80

FIG. 2B

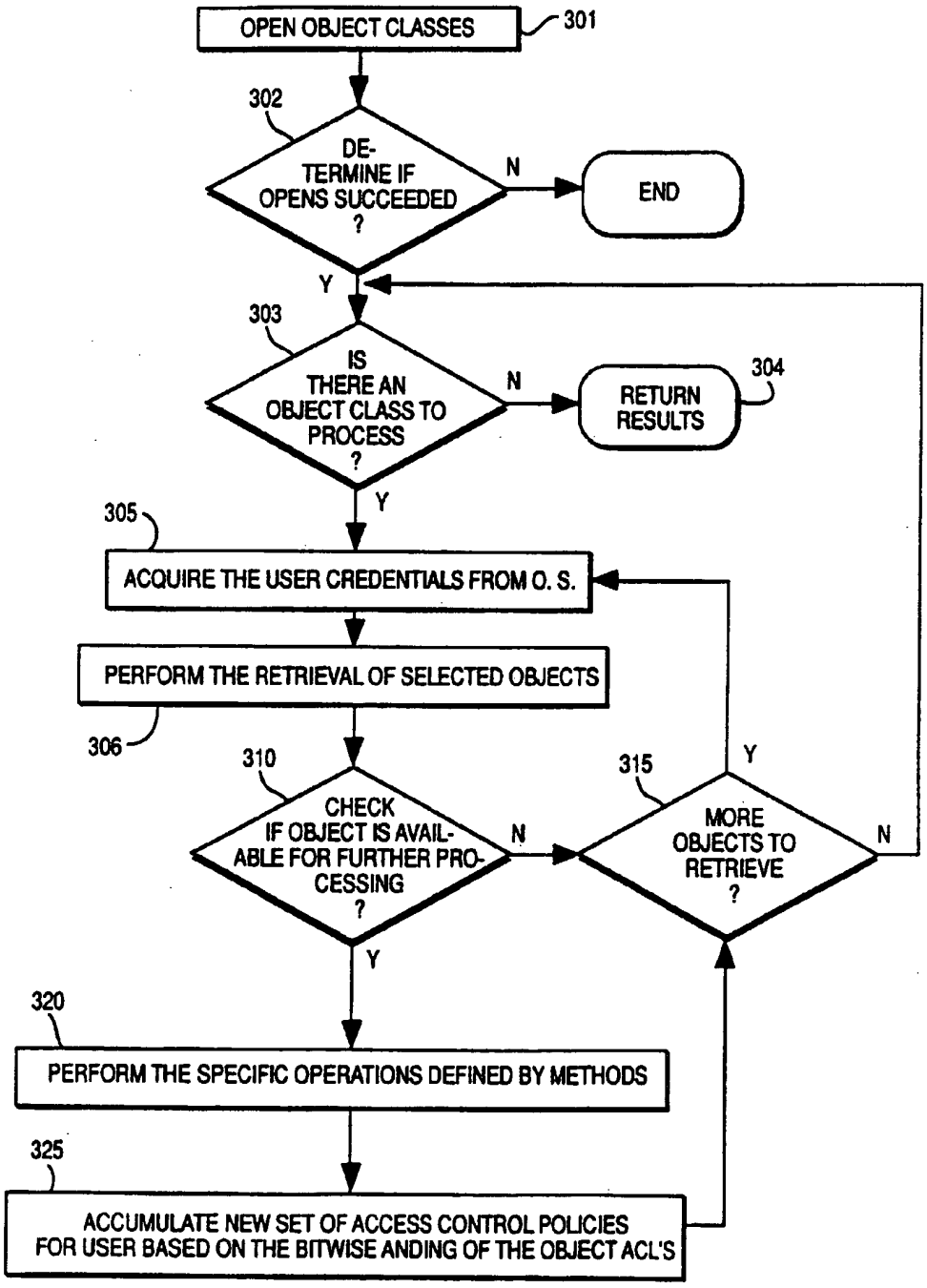
OPEN OBJECT CLASSES
301
302
DE-TERMINE IF OPENS SUCCEEDED ?
N
END
Y
303
IS THERE AN OBJECT CLASS TO PROCESS ?
N
RETURN RESULTS
304
Y
305
ACQUIRE THE USER CREDENTIALS FROM O. S.
PERFORM THE RETRIEVAL OF SELECTED OBJECTS
306
310
CHECK IF OBJECT IS AVAIL-ABLE FOR FURTHER PRO-CESSING ?
N
315
MORE OBJECTS TO RETRIEVE ?
Y
N
Y
320
PERFORM THE SPECIFIC OPERATIONS DEFINED BY METHODS
325
ACCUMULATE NEW SET OF ACCESS CONTROL POLICIES FOR USER BASED ON THE BITWISE ANDING OF THE OBJECT ACL'S

FIG. 3A

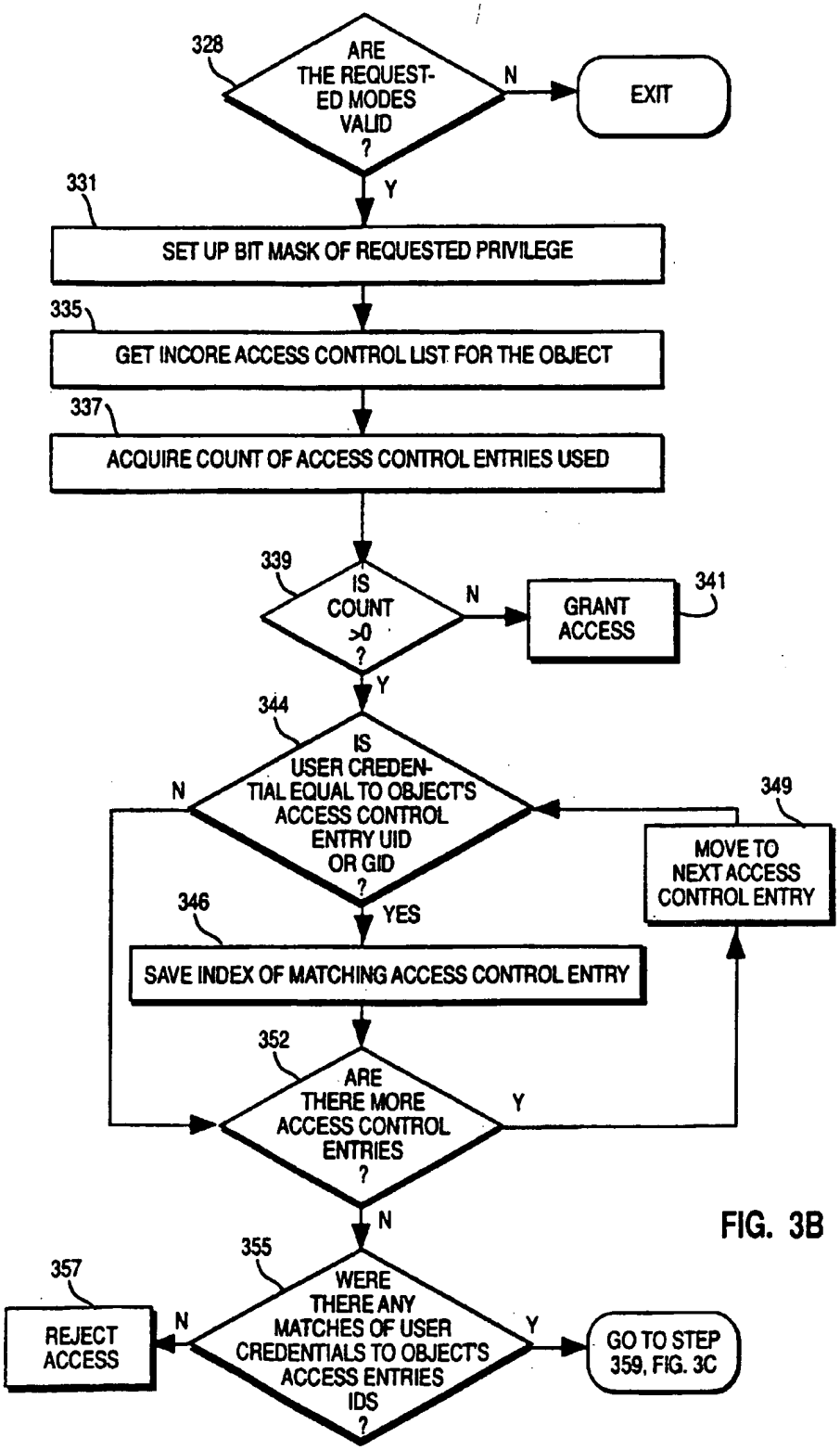
328
ARE THE REQUEST-ED MODES VALID ?
N
EXIT
Y
331
SET UP BIT MASK OF REQUESTED PRIVILEGE
335
GET INCORE ACCESS CONTROL LIST FOR THE OBJECT
337
ACQUIRE COUNT OF ACCESS CONTROL ENTRIES USED
339
IS COUNT >0 ?
N
GRANT ACCESS
341
Y
344
IS USER CREDEN-TIAL EQUAL TO OBJECT'S ACCESS CONTROL ENTRY UID OR GID ?
N
349
MOVE TO NEXT ACCESS CONTROL ENTRY
YES
346
SAVE INDEX OF MATCHING ACCESS CONTROL ENTRY
352
ARE THERE MORE ACCESS CONTROL ENTRIES ?
Y
N
355
WERE THERE ANY MATCHES OF USER CREDENTIALS TO OBJECT'S ACCESS ENTRIES IDS ?
357
REJECT ACCESS
N
Y
GO TO STEP 359, FIG. 3C

FIG. 3B

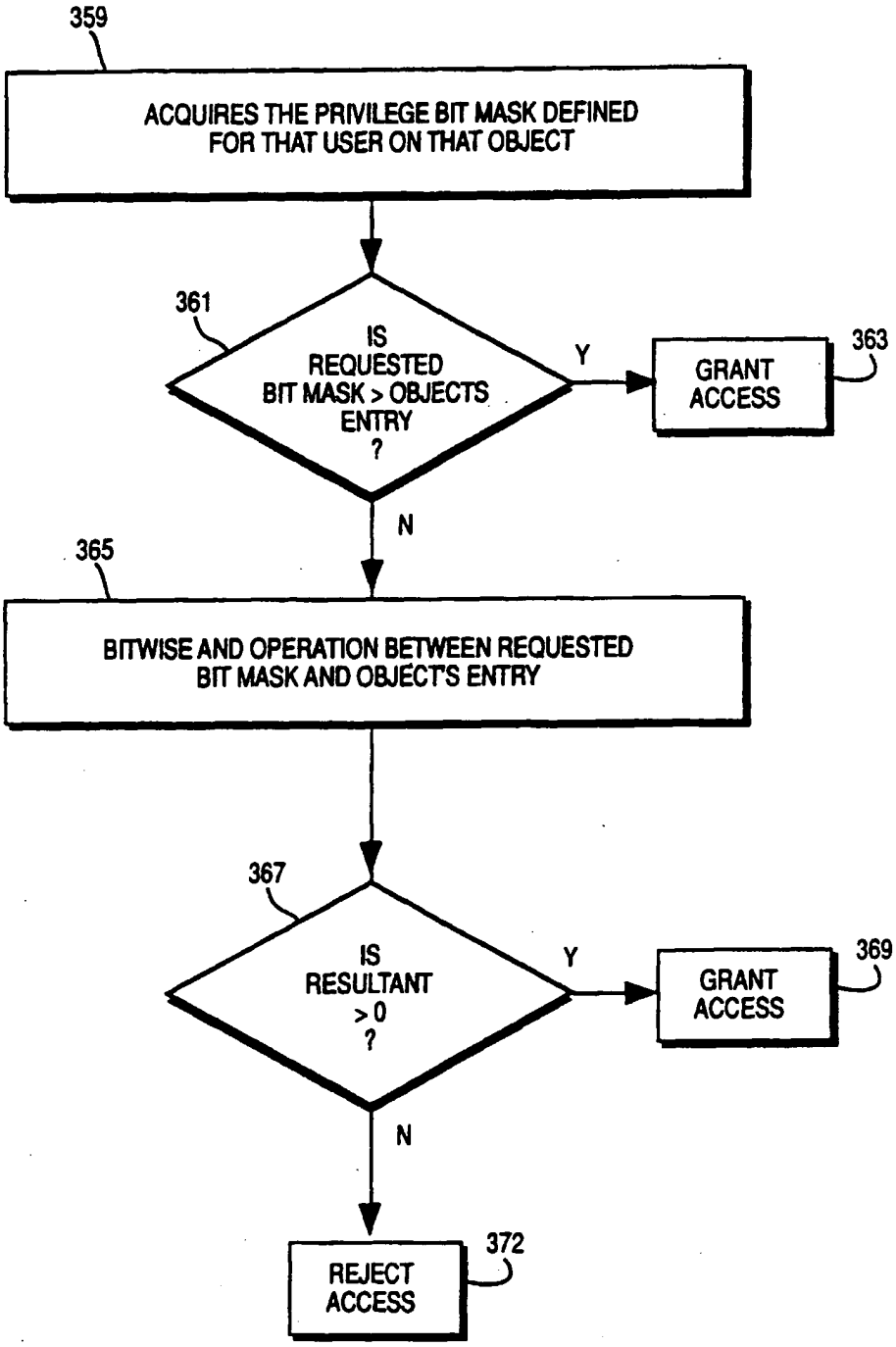
359
ACQUIRES THE PRIVILEGE BIT MASK DEFINED FOR THAT USER ON THAT OBJECT
361
IS REQUESTED BIT MASK > OBJECTS ENTRY ?
Y
363
GRANT ACCESS
N
365
BITWISE AND OPERATION BETWEEN REQUESTED BIT MASK AND OBJECT'S ENTRY
367
IS RESULTANT > 0 ?
Y
369
GRANT ACCESS
N
372
REJECT ACCESS

FIG. 3C

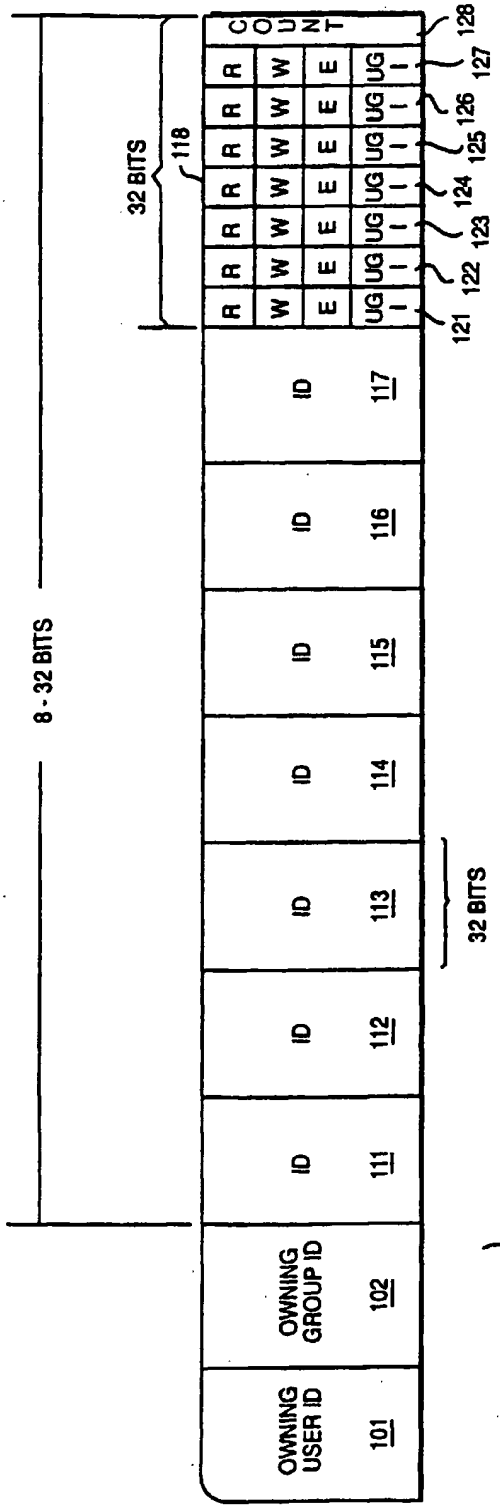
OWNING USER ID
101
OWNING GROUP ID
102
ID
111
ID
112
ID
113
ID
114
ID
115
ID
116
ID
117
8 - 32 BITS
32 BITS
32 BITS
118
R W
R W
R W
R W
R W
R W
R W
R W
E
E
E
E
E
E
E
E
UG
UG
UG
UG
UG
UG
UG
UG
1
1
1
1
1
1
1
1
COUNT
121
122
123
124
125
126
127
128
100

FIG. 4

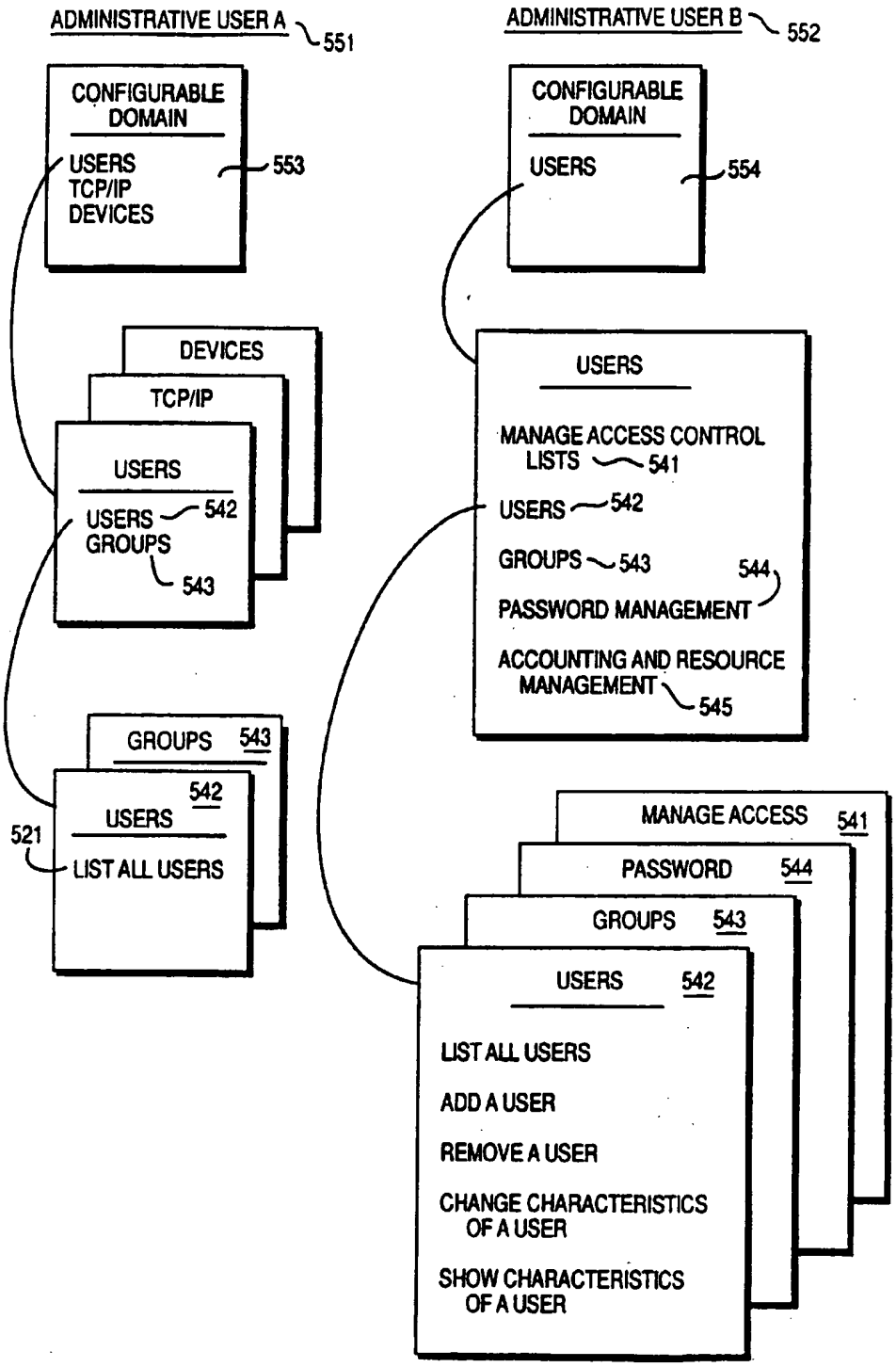
ADMINISTRATIVE USER A
551
CONFIGURABLE DOMAIN
USERS
TCP/IP
DEVICES
553
DEVICES
TCP/IP
USERS
USERS
542
GROUPS
543
GROUPS
543
542
USERS
521
LIST ALL USERS
ADMINISTRATIVE USER B
552
CONFIGURABLE DOMAIN
USERS
554
USERS
MANAGE ACCESS CONTROL LISTS
541
USERS
542
GROUPS
543
544
PASSWORD MANAGEMENT
ACCOUNTING AND RESOURCE MANAGEMENT
545
MANAGE ACCESS
541
PASSWORD
544
GROUPS
543
USERS
542
LIST ALL USERS
ADD A USER
REMOVE A USER
CHANGE CHARACTERISTICS OF A USER
SHOW CHARACTERISTICS OF A USER

FIG. 5A

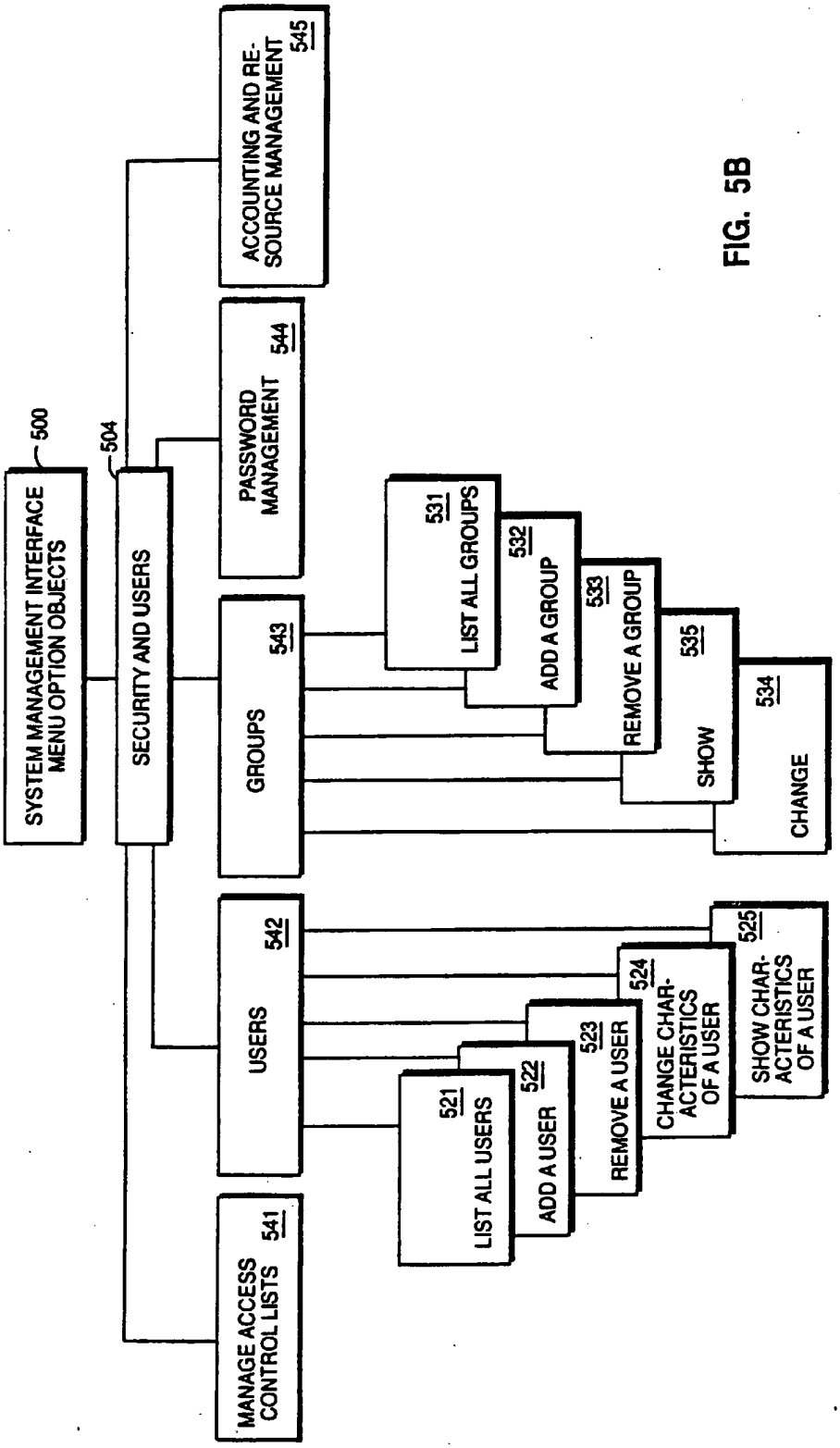
500
SYSTEM MANAGEMENT INTERFACE MENU OPTION OBJECTS
504
SECURITY AND USERS
ACCOUNTING AND RE-SOURCE MANAGEMENT 545
PASSWORD MANAGEMENT 544
GROUPS 543
531 LIST ALL GROUPS
532 ADD A GROUP
533 REMOVE A GROUP
535 SHOW
534 CHANGE
USERS 542
521 LIST ALL USERS
522 ADD A USER
523 REMOVE A USER
524 CHANGE CHAR-ACTERISTICS OF A USER
525 SHOW CHAR-ACTERISTICS OF A USER
MANAGE ACCESS CONTROL LISTS 541

FIG. 5B